Taller Guiado - Análisis de Algoritmos

Alberto Luis Vigna Arroyo, Camilo José Martinez

30 de enero de 2024

Abstract

En este documento se presenta el análisis del algoritmo de ordenamiento Bubble Sort, Insertion Sort y Quick Sort. Cada uno de estos algoritmos aborda el problema de ordenar una lista de elementos, pero difieren en sus enfoques y eficiencias. A continuación, se proporciona un resumen de los aspectos más destacados de cada algoritmo, destacando sus características, ventajas y desventajas, así como sus complejidades temporales y espaciales. Este análisis permite comprender mejor la idoneidad y el rendimiento de cada algoritmo en diferentes contextos y escenarios.

Part I
Análisis y diseño del problema

1 Análisis

El problema, informalmente, se puede describir como calcular para una secuencia de elementos $S = \langle a_1, a_2, a_3, \cdots, a_n \rangle$ en donde $\forall_i a_i \in \mathbb{T}$ y $\mathbb{T}$ existe una relación de orden parcial $' \leq'$, una secuencia ordenada, es decir una permutación $S = \langle a'_1, a'_2, a'_3, \cdots, a'_n \rangle$ en donde $a_i \leq a'_{i+1}$ y $a'_i \in S$.

2 Diseño

Con las observaciones presentadas en el análisis anterior, podemos escribir el diseño de un algoritmo que solucione el problema . A veces este diseño se conoce como el «contrato» del algoritmmos o las «precondiciones» y «poscondiciones» del algoritmo. El diseño se compone de entradas y salidas:

Definition. Entradas:

1. Definición entrada 1
2. Definición entrada 2

Definición. Salidas:

1. Definición salida 1
2. Definición salida 2

Parte II
Algoritmos

3 Opción algoritmo 1 - BubbleSort

3.1 Algoritmo

Este algoritmo se basa en el principio de comparar pares de elementos adyacentes en una lista y realizar intercambios si es necesario, repitiendo este proceso hasta que la lista esté completamente ordenada. En cada iteración se va recorriendo la lista y verificando si el elementos $a_i > a_{i+1}$. Si este lo es los elementos se "rotan" para que estos vayan siendo ordenados.

Algorithm 1 BubbleSort

```
1:  procedure BubbleSort(S)
2:      n ← |S|
3:      for i ← 0 to n − 1 do
4:          for j ← 0 to n − i − 1 do
5:              if S[j] > S[j + 1] then
6:                  swap S[j], S[j + 1]
7:              end if
8:          end for
9:      end for
10: end procedure
```

3.2 Complejidad

El algoritmo X tiene órden de complejidad . ¿Por qué?

3.3 Invariante

- **Inicio**:
- **Avance**:
- **Terminación**

3.4 Notas de implementación

Parte III
Comparación de los algoritmos